IN-61-CR
43119
P-33

An Intelligent Tutoring System for the Investigation of High Performance Skill Acquisition

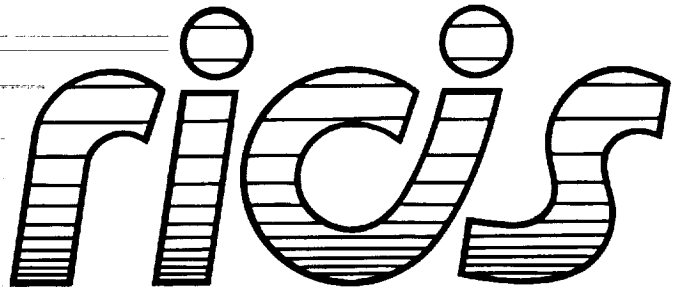
Pamela K. Fink
L. Tandy Herren
Southwest Research Institute

J. Wesley Regian
Armstrong Laboratory Human Resources Directorate

May 1991

Cooperative Agreement NCC 9-16
Research Activity No. ET.23

NASA Johnson Space Center
Mission Operations Directorate
Space Station Training Office



Research Institute for Computing and Information Systems
University of Houston - Clear Lake

T·E·C·H·N·I·C·A·L R·E·P·O·R·T

The RICIS Concept

The University of Houston-Clear Lake established the Research Institute for Computing and Information systems in 1986 to encourage NASA Johnson Space Center and local industry to actively support research in the computing and information sciences. As part of this endeavor, UH-Clear Lake proposed a partnership with JSC to jointly define and manage an integrated program of research in advanced data processing technology needed for JSC's main missions, including administrative, engineering and science responsibilities. JSC agreed and entered into a three-year cooperative agreement with UH-Clear Lake beginning in May, 1986, to jointly plan and execute such research through RICIS. Additionally, under Cooperative Agreement NCC 9-16, computing and educational facilities are shared by the two institutions to conduct the research.

The mission of RICIS is to conduct, coordinate and disseminate research on computing and information systems among researchers, sponsors and users from UH-Clear Lake, NASA/JSC, and other research organizations. Within UH-Clear Lake, the mission is being implemented through interdisciplinary involvement of faculty and students from each of the four schools: Business, Education, Human Sciences and Humanities, and Natural and Applied Sciences.

Other research organizations are involved via the "gateway" concept. UH-Clear Lake establishes relationships with other universities and research organizations, having common research interests, to provide additional sources of expertise to conduct needed research.

A major role of RICIS is to find the best match of sponsors, researchers and research objectives to advance knowledge in the computing and information sciences. Working jointly with NASA/JSC, RICIS advises on research needs, recommends principals for conducting the research, provides technical and administrative support to coordinate the research, and integrates technical results into the cooperative goals of UH-Clear Lake and NASA/JSC.

An Intelligent Tutoring System for the Investigation of High Performance Skill Acquisition

[redacted] INTENTIONALLY [redacted]

Preface

This research was conducted under auspices of the Research Institute for Computing and Information Systems by Dr. Pamela K. Fink and Dr. L. Tandy Herren of the Southwest Research Institute and Dr. J. Wesley Regian of the Armstrong Laboratory Human Resources Directorate, Brooks Air Force Base, Texas. Dr. Glenn Freedman, Director of the Software Engineering Professional Education Center at the University of Houston-Clear Lake, served as RICIS research coordinator.

Funding has been provided by the Mission Operations Directorate, NASA/JSC through Cooperative Agreement NCC 9-16 between the NASA Johnson Space Center and the University of Houston-Clear Lake. The NASA technical monitor for this activity was Barbara N. Pearson of the Systems/Elements Office, Space Station Training Office, Mission Operations Directorate, NASA/JSC.

The views and conclusions contained in this report are those of the authors and should not be interpreted as representative of the official policies, either express or implied, of NASA or the United States Government.

Contract No. NASA NCC-9-16
Subcontract No. 060
RICIS Research Activity No ET.23
SwRI Project No. 05-3515

AN INTELLIGENT TUTORING SYSTEM FOR THE INVESTIGATION OF HIGH PERFORMANCE SKILL ACQUISITION

Pamela K. Fink, Ph.D.
L. Tandy Herren, Ph.D.

Southwest Research Institute
San Antonio, Texas 78228-0510

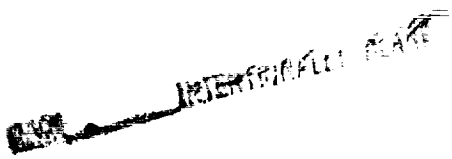
J. Wesley Regian, Ph.D.

Armstrong Laboratory Human Resources Directorate
Brooks AFB, Texas 78235-5601

ABSTRACT

The issue of training high performance skills is of increasing concern. These skills include tasks such as driving a car, playing the piano, and flying an aircraft. Traditionally, the training of high performance skills has been accomplished through the use of expensive, high-fidelity, 3-D simulators and/or on-the-job training using the actual equipment. Such an approach to training is quite expensive. This paper describes the design, implementation, and deployment of an intelligent tutoring system developed for the purpose of studying the effectiveness of skill acquisition using lower-cost, lower-physical-fidelity, 2-D simulation. Preliminary experimental results are quite encouraging, indicating that intelligent tutoring systems are a cost-effective means of training high performance skills.

Support for this research was provided under RICIS Research Activity No. ET.5 (NASA Cooperative Agreement NCC 9-16) through funding from the Air Force's Armstrong Laboratory Human Resources Directorate.



1 INTRODUCTION

The Console Operations Tutor was fielded to the Armstrong Laboratory Human Resources Directorate (HRD) at Brooks Air Force Base and to NASA/Johnson Space Center in June of 1990. The primary goal of this system was not to field another intelligent tutoring system, but rather to develop a psychological research platform for investigating issues in training a class of tasks referred to as "high performance." High performance skills are physically-oriented tasks that can be performed without much cognitive elaboration (Regian and Shute, 1988). Driving a car is a high performance task and, while initially driving requires a high degree of concentration, with practice the skill no longer consumes much thought. Development of such an intelligent tutoring system, where artificial intelligence-based techniques are used to design and implement a computer system to support research in human learning, is a natural and important means of extending the interdisciplinary approach that is essential to progress in both fields, artificial intelligence and human psychology.

The Console Operations Tutor has been used extensively as a research tool at the Armstrong Laboratory Human Resources Directorate, Brooks AFB and at NASA/Johnson Space Center. The results suggest that a modified apprenticeship strategy for training high performance skills effectively allows for transfer of the skill from a 2-D computer environment to the actual 3-D device. This indicates that less expensive, 2-D tutoring systems can be used to deliver some of the training currently performed on large, expensive, high fidelity, 3-D simulators. The following sections review the rationale for the development of, and approach used in, the Console Operations Tutor, including the instructional strategy used by the tutoring system, the components of the tutoring system, and the results of fielding the system to conduct psychological research.

1.1 The Need For ICAI Systems

Technological advances during the last several decades have resulted in the automation of many menial tasks. As a result, need has arisen for a more skilled workforce capable of running, monitoring, and maintaining these automated systems. Though such jobs may be based on a general knowledge of some problem solving area, such as fluid flow or electronics, the tasks that must be performed hinge on very specific knowledge about the specific processes and devices employed at a given location. As a result, though an individual may have obtained an education in some engineering discipline, a need still exists for very specialized training in the given job tasks. To further complicate the training picture, the average American worker is changing jobs more frequently than in the past. With each job change, there is potentially a need for different, specialized training.

Job proficiency often requires a wide range of knowledge and skill types, including declarative knowledge, procedural knowledge, and psychomotor skill. For example, to monitor a complex system, an individual needs to have a general understanding of the principles behind the

functioning of the system as well as specific knowledge of the particular device. In addition, the individual needs to have the skills necessary for manipulating and controlling the devices used to collect information on the performance of the system. Thus, both cognitively-oriented knowledge and physically-oriented skills are required for job proficiency. A large literature exists regarding which instructional approaches are optimal for the various categories of knowledge and skill.

Traditionally, training of knowledge-intensive tasks has used an approach that relies on lectures and reading. A single human instructor, a number of students, and a set of written material, one set for each student, is a typical classroom scenario. The students listen to lectures, read books and documents, ask questions of the teacher, and pass a more or less verbally-oriented exam at the end to demonstrate proficiency. This is the training approach that has been used in the traditional education system for centuries. It is also the approach used for many of the cognitively-oriented tasks that such organizations as NASA and the U.S. Air Force must train.

Training of physically-oriented skills has traditionally been approached through apprenticeships, high-fidelity, three-dimensional mock-ups, and/or on-the-job training. Students may first acquire some preliminary knowledge about the task(s) from a lecture or reading material, but the ultimate training is actual execution of the task, either in the real or a simulated environment. The student is repeatedly run through trials on the task, monitored and directed as-needed by a human instructor, until the required level of proficiency is attained. This is the training approach that has been used throughout history for trades such as silversmithery and stonemasonry. It is also the methodology employed by NASA and the U.S. Air Force for training the physically-oriented tasks performed by astronauts and aircraft pilots.

A key attribute of both of these approaches to training is their human-intensive nature. No matter what knowledge or skill is trained, a human instructor is the key element to the process. The idea that teaching is best accomplished by tailoring instruction to individual students is both ancient and ubiquitous among instructional threorists. Corno and Snow (1985) found the idea detailed in the fourth century B.C. Chinese Xue Ji, in the ancient Hebrew Haggaday of Passover, and in the first century Roman De Institutione Oratoria. Today the basic idea still forms the core of several important streams of research on instruction. The promise of individualized instruction is the basis of research on mastery learning (e.g., Bloom, 1956; Carroll, 1963; Cohen, Kulik, and Kulik, 1982), aptitude-treatment interactions (e.g., Corno and Snow, 1985; Cronback and Snow, 1977; Shute, in press), apprenticeship learning (Gott, 1988; Collins, Brown, and Newman, 1987), and intelligent tutoring systems (e.g., Sleeman and Brown, 1982; Lewis, McArthur, Stasz, and Zmuidzinas,1990, Woolf, 1987). The idea also has strong empirical support. A consistent finding is that when using traditional stand-up instruction, other things being equal, smaller class sizes produce superior learning outcomes (Bloom, 1984). The most common interpretation of this result is that smaller classes enable instructors to be more aware of, and responsive to, the needs of individual students.

This human-intensive approach to training makes it very expensive.

Organizations such as the U.S. Air Force and NASA, as well as private industry, can not afford such expense, especially when coupled with increasing training demands. As a result of the increasing expense and the improvement in computer capabilities, the past couple of decades have seen an evolution in computer-aided instruction (CAI). CAI does not remove the human instructor from the training loop, but rather it seeks to enhance the human instructor and reduce the instructor's workload. At the same time, it provides consistent, appropriate, and individualized instruction.

1.2 Development Of ICAI Systems

Initially, CAI systems were nothing more than computer-operated page-turners (Arons, 1984). Then systems basically allowed the student to read text on a computer screen, instead of in a book, on a self-paced basis. Over time, conventional CAI expanded into the use of conditional branching (based on student performance) to individualize the instruction, and often included realistic, visually-oriented simulations. These simulations could be either two-dimensional, using technologies such as computer graphics, interactive videodisk (IVD), or digital video interactive (DVI), or three dimensional, relying predominantly on large, high-fidelity, three-dimensional mock-ups of the environment to be trained. The courseware for such systems is still designed and administered, and student performance is still evaluated, by human instructors in order to determine the next step in the training process. Thus, conventional CAI has removed some of the workload from the instructor, while still providing students with many of the benefits of one-on-one teaching.

Intelligent computer-aided instruction (ICAI) takes the individualized instruction one step further, providing the computer system with more adaptive and flexible responses to student performance. In addition, ICAI systems, often referred to as intelligent tutoring systems (ITS's) (Sleeman and Brown, 1982), maintain an explicit model of the expertise to be trained, of the student being trained, and of the instructional strategy employed in providing trials, feedback, and remediation to the student. ICAI systems involve the encoding of the knowledge on which decisions concerning a student's training session should proceed, rather than the explicit encoding of the decisions themselves (Wenger, 1987). These "intelligent" training systems have usually been developed for cognitively-oriented tasks, such as Anderson's Lisp Tutor (Anderson, Farell, and Sauers, 1984), Brown and Burton's SOPHIE system for electronic diagnosis (Brown, Burton, and deKleer, 1982), Carbonell and Collins' SCHOLAR system for South American geography (Carbonell, 1970), and Woolf and McDonald's MENO-TUTOR for diagnosing non-syntactic bugs in computer programs (Woolf and McDonald, 1985).

As a result, ICAI, or ITS's, can be regarded from the viewpoint of degree of individualization in training as simply an evolutionary change from the original CAI work (Regian, 1989). However, from the viewpoint, of designing and implementing software systems, ICAI systems appear revolutionary (Dede and Swigger 1987). In order to obtain the individualized instruction, a different approach to computer programming must be used. In conventional programming for CAI systems, the actual

decisions are encoded, while in AI-based programming for ITS's the knowledge required to make those decisions is encoded. The evolutionary aspects of a move from CAI to ICAI allow one to conceive of building an ITS to support training of the more traditional problem solving tasks normally approached using conventional instructional systems design strategies and CAI. It is the revolutionary aspects that make actual design and implementation of such systems more complex than might be expected initially.

The following paper describes the development and fielding of a computer-based training system for a physically-based, skill-oriented task. This system is referred to as the Console Operations Tutor and is unique for several reasons. First, an AI-based approach was used in the design and implementation of the system, yet the problem solving domain to be trained, namely a high performance, skill-oriented task, has more traditionally been approached using conventional CAI. Furthermore, the goal of system development was not so much to generate a tutoring system for training a set of students targeted for a specific job slot, but rather to develop a research tool for investigating whether or not the area of intelligent tutoring systems is an effective means of training a particular class of high performance skills. Effectiveness is defined as lower cost, shorter time-to-train, and longer retention of the skill. Results of this research will provide direction to individuals in a position to define training needs and to individuals involved in designing and implementing training systems.

2 DEVELOPMENT OF AN ITS FOR A HIGH PERFORMANCE TASK

The intelligent tutoring system arena is a research area owned, to some extent, by both the psychology and the computer science fields. It is a meeting point for educational/learning theory in humans and design and implementation of complex computer software that exhibits intelligent behavior. As a result, the design and development of intelligent tutoring systems is a highly interdisciplinary endeavor. Top level design issues such as characterization of the knowledge to be taught and the appropriate approach or approaches to teaching this knowledge require an understanding of psychology, education, computer science, human factors, and the subject matter domain.

A standard accepted architecture for an ITS is presented in Figure 1. The major modules include a expert model, an instructional expert, a student model, and an intelligent interface. Though earlier ITS efforts tried to develop these major modules separately, experience has shown that the knowledge contained in these modules is highly interrelated and that performing each of the functions of an ITS requires knowledge available in more than one module at a time. For example, instructional knowledge tends to be embodied in the student model and intelligent interface, as well as in the instructional module. Domain expertise resides in the student interface and the expert model, which may together provide a simulation facility.

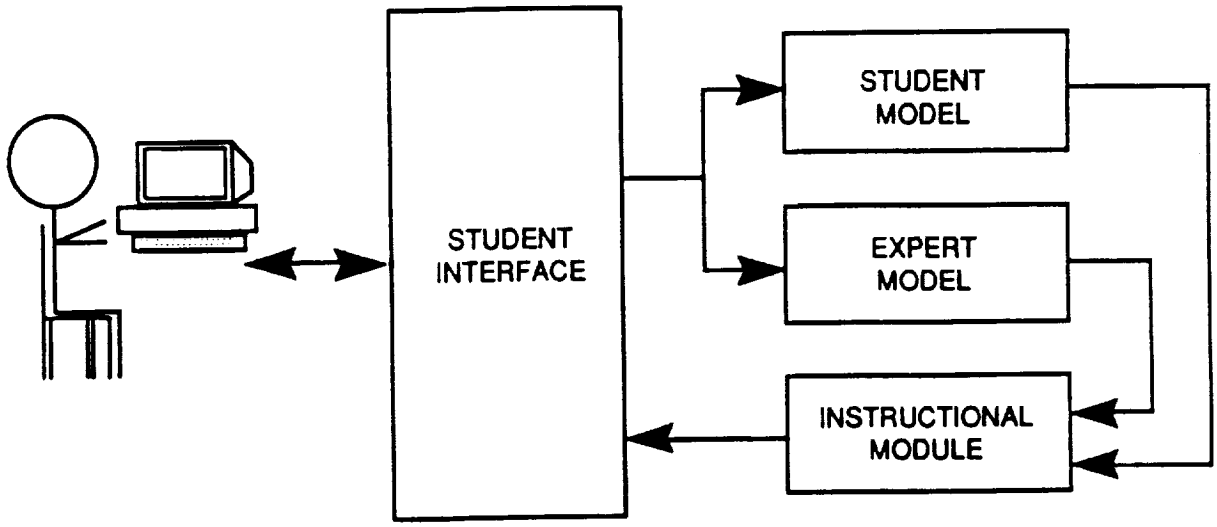


Figure 1: An Architecture for an Intelligent Tutoring System

From a software development perspective, each of the major modules is fairly complicated in its own right, and their inherent interrelationships complicate ITS development further. Also, the kind of knowledge that must be encoded can be quite complex. From a psychology perspective, issues in education and learning remain open research topics and what constitutes effective teaching for a domain is not always fully understood. But, this kind of knowledge must be used as the foundation of the instructional module for any effective ITS, and it essentially constitutes the design and implementation of an expert system for teaching the particular domain.

A further complication in ITS development is that domain expertise of the appropriate kind and level for teaching must be implemented in the expert model to be used by the instructional portion of the system for training. This constitutes the design and implementation of a specialized knowledge-based system that contains the knowledge and problem solving skills that must be taught to the student. The development of an ITS is equivalent to the development of several very different expert systems that all must work together, requiring extensive effort on the part of experts from all of the fields concerned.

As a result of this dual nature of intelligent tutoring systems, the following sections discuss the Console Operations Tutor from two different perspectives. The first is from the psychology and education perspective and addresses the instructional model used to drive the system. The second is the from the computer science and artificial intelligence perspective, which deals with the software design used to implement the tutoring system.

2.1 An Instructional Strategy For Teaching High Performance Skills

Human performance in almost any cognitive or motor skill shows profound changes with practice. The law of practice is ubiquitous in cognitive performance domains (Newell and Rosenbloom, 1981). Consider the changes that occur while learning to fly an aircraft, type, play a musical instrument, read, or play tennis. At first, effort and attention must be devoted to every movement or minor decision. At this stage, performance is slow and error prone. Eventually, complex tasks can be carried out with little attention, and performance is quite rapid and accurate. For example, in aircraft control, the novice may have difficulty just keeping the aircraft on the proper heading. However, the expert can fly complex aircraft formation maneuvers while performing a simultaneous digit cancelling task (Colle and DeMaio, 1978).

There are other examples of skills for which performance does not increase with practice. For example, measures of short term memory capacity such as memory scanning rate for comparing random symbols (Kristofferson, 1972) or working memory capacity (Chase and Ericson, 1981) are insensitive to practice. The automatic/controlled processing framework (Schneider and Shiffrin, 1977; Shiffrin and Schneider 1977) provides a tool for distinguishing between trainable and untrainable skills.

The framework posits two qualitatively different forms of processing that underlie human performance. Automatic processing is fast, parallel, fairly effortless, not limited by short term memory capacity, not under direct subject control, and is used in performing well-developed skilled behaviors. This mode of processing develops when subjects perform in a consistent manner over many trials. Controlled processing is slow, effortful, cognitive capacity-limited, subject-controlled, and is used to deal with novel, inconsistent, or poorly learned information. This mode of processing is expected at the beginning of practice on any novel task. In this framework, trainable skills are trainable because they involve components that can be automatized. Automatized components are executed rapidly, reliably, and with little effort, freeing capacity for performing other non-automatic task components.

In designing training procedures for procedural skills, three important findings from the automatic/controlled processing framework should be considered. The first centers on the distinction between consistent practice and varied (or inconsistent) practice. Consistent practice produces substantial improvements in performance as automatic processing develops (e.g., 98% reduction in visual search comparison rates, Fisk and Schneider, 1983). Varied practice uses only controlled processing and produces little improvement in performance (e.g., no change in letter search performance over 4 months of training, Shiffrin and Schneider, 1977). The second finding centers on the amount of effort required to perform automatic processing tasks. Consistent practice greatly reduces the amount of effort required to perform a task, allowing controlled processing to be allocated to another task. When subjects have already developed automatic processes to perform one task, they can learn to time-share another task with little or no deficit. After 20 hours of consistent practice in two search tasks, subjects were able to perform both tasks simultaneously nearly as well as they could perform each separately

(Fisk and Schneider, 1983). The third finding is that automatized performance is far more reliable under stress (see Hancock and Pierce, 1984) than non-automatized performance. These findings indicate that a training program that includes consistent training for a skill will provide performance that is resistant to degredation from concurrent tasks or an individual's emotional state.

According to a commonly accepted model of skill learning (Anderson, 1983), skill acquisition can be seen as a progression that begins with the encoding of declarative (factual) knowledge, continues with learning of procedural (action-centered) knowledge/skill, and with sufficient practice leads to the acquisition of automatic (cognitively automatized) skill. That is, skills are initially acquired as declarative knowledge, the knowledge is then proceduralized into 'action recipes', and these action recipes finally become cognitively automatized.

Progression through the three stages of skill acquisition can be monitored with three classes of performance measures: accuracy, speed, and resource load. Early on, during declarative knowledge acquisition, task accuracy improves rapidly to some asymptotic level. Next, during procedural knowledge/skill acquisition, task latency gradually declines to some limiting level. Finally, attentional resource requirements for the target task begin to shrink. This reduction in attentional resource requirement is measured by having the trainee perform a carefully-designed secondary task while concurrently performing the target task. When the trainee is capable of performing both tasks concurrently without decrementing performance latency or accuracy, the target task is said to be automatized. Figure 2 shows an idealized strip-chart representation of the three performance measures as they change with practice.

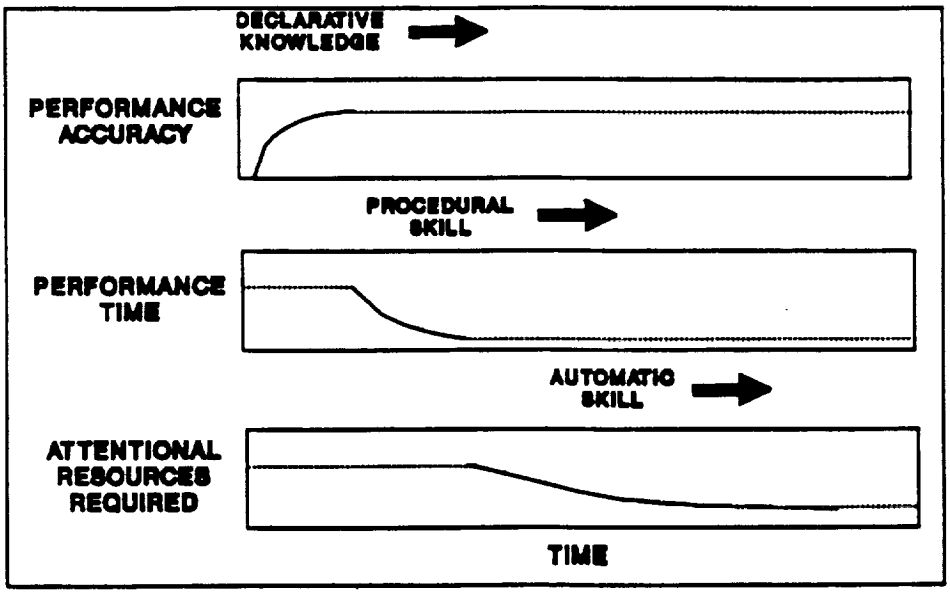


Figure 2: Accuracy, Speed, and Resource Load Measures of Automatized Skills

2.2 Overview Of The Instructional Strategy Used In The Console Operations Tutor

Our instructional strategy capitalizes on these hallmarks of high performance skill acquisition. Early in training, we focus on subjects learning accurate declarative knowledge, without regard to peformance speed. After the student has reached a desired level of accuracy, the focus shifts to performance speed. When the student has reached a desired level of performance speed while still maintaining accuracy, the focus shifts to performance automaticity. Table 1 summarizes the relationship between the sequential instructional approaches implemented in the tutor and the knowledge/skill types.

The static overview phase begins with a guided tour of the system. It starts at a large grain, detailing what major systems or groups of systems exist. It works its way down to a finer level of detail until all relevant entities have been learned. At each level of detail, systems or groups of systems are highlighted while the trainee is told what the system is and what its functions are. The static overview is terminated with a recognition test.

In the procedural instruction phases, the trainee is taught the procedures in three passes. In the first pass (general procedure), the trainee is shown the procedures and the various steps are explained. In the second pass (guided example), the student is required only to perform the procedure in the correct order. The software coach prompts the student for the next step until the procedure is learned. When the trainee can do the procedure in the correct order on a criterion number of consecutive occasions without prompting, s/he proceeds to speed training. In speed training (unguided example), the trainee's latency is monitored, and the trainee is given feedback on their speed.

In automaticity training (automated example), the system uses a secondary task to enhance and diagnose cognitive automaticity. In the secondary task, the student hears a series of tones and is told, for example, on a given trial to respond in one manner if s/he hears two shorts followed by a long and in another manner if s/he hears two longs followed by a short. The goal of automaticity training is to have the trainee be able to perform the primary task (the procedure) in the presence of a secondary task, with no performance decrement on the primary task while maintaining accurate performance on the secondary task.

Instructional Approach Phases	Resulting Knowledge/Skill
Static Overview Knowledge	Declarative Knowledge
General Procedure-Oriented Knowledge	Procedural Knowledge
Guided-Example Exercises	Procedural Knowledge/Skill
Unguided-Example Exercises	Procedural Skill
Automated-Example Exercises	Automatized Skill

Table 1: Instructional Approaches and Resulting Knowledge/Skill Types

This training process moves the student gradually through the various phases of skill development. How long a student might spend at any given level of training is, of course, unknown. It depends to some degree on the difficulty of the domain as well as the individual student's capabilities. In the last three phases of training, where skill acquisition is taking place, no set number of trials, level of accuracy, or rate of performance improvement is predetermined by the instructional module of the ITS. These vary based on student aptitude and the domain complexity. When to move on to the next level of training versus when to remain at the same level or even backup and remediate, is based on the individual student's performance. Some students may never perform as well on the task as other students.

For the tutoring system to be intelligent, it must be capable of recognizing when a student has "peaked" on a particular training phase. The goal is to determine how skillful the student is based on actual performance, not on how many times they have done the procedure. This requires a decision on how accurate is accurate enough and how fast is fast enough at each level of training to indicate that the student has acquired a proficiency at that level and is ready to move on to the next level of training. These are issues in which AI-based techniques provide the training system with the knowledge to make these decisions on a case-by-case basis, thus moving toward a more adaptive, individually responsive, and intelligent tutoring system.

2.3 The AI-Based Design For A Tutoring System To Teach High Performance Skills

From an AI perspective, several issues arise concerning how various types of knowledge should be represented in the system and how to provide as general a training tool as possible to serve both as a psychological research tool and a viable training system (see Fink, in press). The tutoring system was initially broken down into the major components of an ITS, in order to simplify the design process. The breakdown organized questions about design into four major categories corresponding to the four major components of the system. The user interface provided a 2-D mock-up of the console and simulated console interactions through a mouse. Further details of this portion of the system are provided in Section 3.2, where interaction with the tutor is described. The instructional strategy described above was implemented in a general way so that additional curriculum could be added, if desired. The expertise within the domain, namely MSK manipulation, is highly procedural and physically-oriented. It is not the sort of expertise normally represented to a computer in an intelligent system, thus a general representation scheme was developed to handle a curriculum for teaching a high performance task and procedure-oriented expertise. From these two representations a student model was developed. The instructional module and the expert and student models are described below.

If the training domain consists of only one specific task to be trained, the five phases of high performance training described in the previous section are easily represented as a directed graph (see Figure

3A). In this representation, leaf nodes correspond to the five phases of skill acquisition, and higher level nodes govern the order in which the training progresses or remediates. Traversing the arcs represents advancement to the next training phase or remediation to a previous phase, depending on the direction.

The console operations domain consists of multiple related tasks, each of which the student must learn to automaticity. In this domain some training phases apply to several or all of the tasks at once, represented by a single leaf node (or lesson) for that training phase (as in Figure 3A). In other phases, training is more effective for one task at a time, represented by multiple leaf nodes for a given phase, one for each specific task. This is shown in Figure 3B, where phases 2 and 3 are split over several lessons, each of which concentrates on training a specific level of learning for one specific task.

In general, training progresses in a left-to-right, depth-first manner through the training tree. This training tree defines the curriculum for the skill set. To fully specify the training order for a domain which comprises several related tasks, it is useful to divide all non-leaf nodes in the tree into two types: sequence nodes and selection nodes. The use of sequence versus selection nodes provides a means of distinguishing between dependent and independent tasks in a training sequence. This is particularly helpful when trying to determine on what item(s) remediation should take place when a student is having difficulty. Leaf nodes in this training tree represent lesson types, such as a static overview or a set of exercises, that will train a particular skill. They refer to structures in the expert model of the tutoring system. This training tree paradigm provides a very general way to represent a curriculum for teaching a specific task or set of tasks.

Sequence nodes (those not marked with black diamonds in Figure 3) cause the training to progress in a sequential manner from their leftmost child node to their rightmost child node. If remediation is indicated while training is at a child of a sequence node, this remediation can be thought of as a traversal up through the parent node and down to the closest sibling to the left. Thus, the term "sequence" indicates that the items to be taught have a temporal relationship in the sense that the ones appearing further to the right in the training tree depend directly on the skills acquired in those appearing to their left.

While sequence nodes govern the advancement and remediation of a student between the five training phases, selection nodes are necessary to control the progress of the student through the training of independent tasks. Selection nodes (marked with black diamonds in the diagram in Figure 3) have child nodes which correspond to independent tasks to be learned. Thus, these nodes are used to represent tasks that are logically independent of one another. No task below a selection node depends upon any other task appearing below that selection node. These nodes behave differently depending on whether or not the current level of training is at the "training frontier," the farthest point in the training traversal that this particular student has ever reached. If the current training level is at the frontier, the children of a selection node are visited in order from left to right, just as with a sequence node. However, if one of these children causes a remediation, then the target of remediation should not be

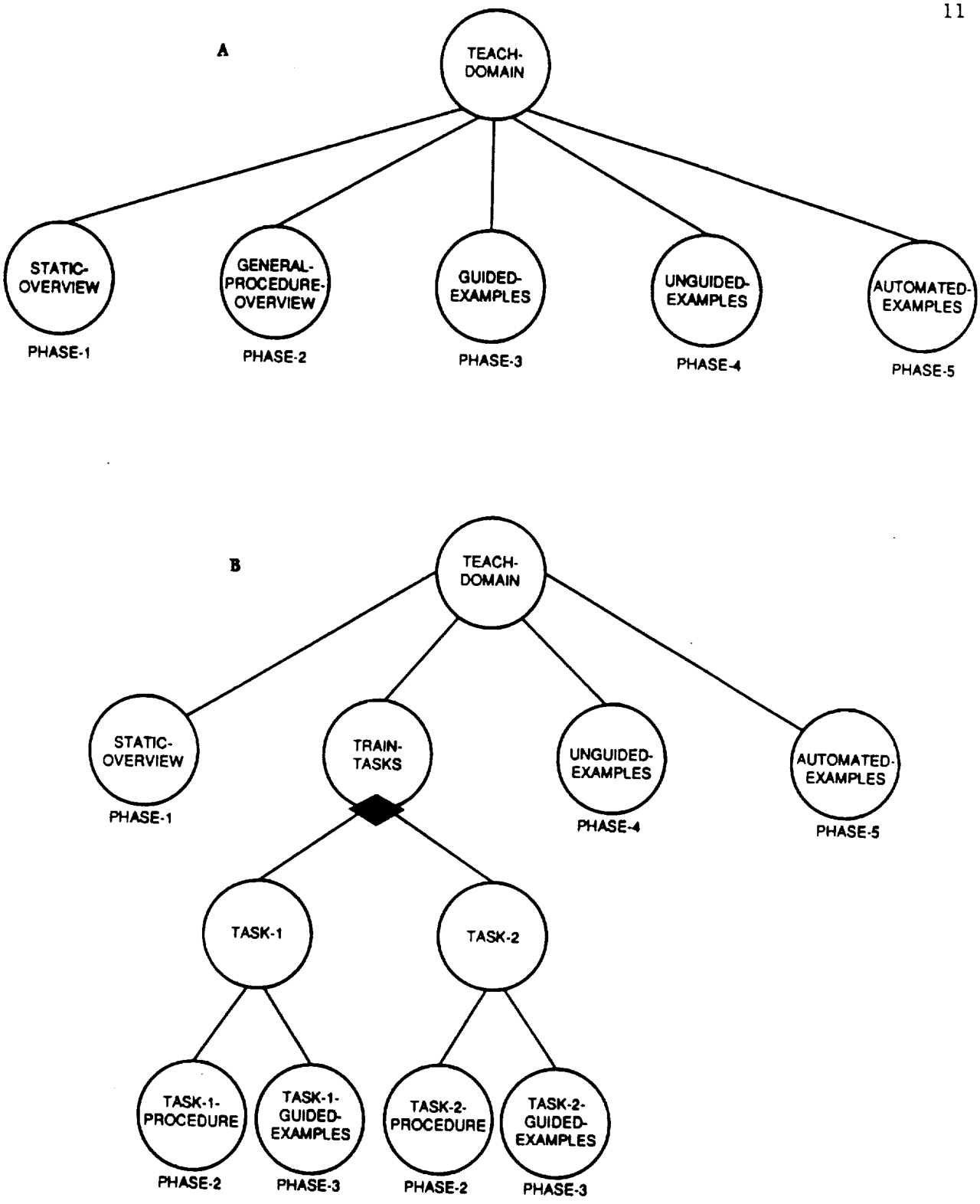


Figure 3: A Single Task Trained Through All Five Phases to Automaticity (A) and a Multi-Task Domain Where Two Independent Subtasks Are Trained Separately in the General-Procedure and Guided-Examples Phases (B)

the previous child of the selection node parent, as this child represents a task which is independent of the one with which the student is having difficulty. The correct target of this remediation is the previous child node of the nearest sequence node in a traversal up the tree from the child. For example, in Figure 3B, if node task-2-procedure causes a remediation, then node static-overview should be the one that is revisited. When the student has again shown mastery of static-overview, then the training must return to task-2-procedure, without re-traversing the task-1-procedure path.

Nodes which are to the right of a selection node in the training tree are cumulative training nodes, presenting exercises for all of the tasks which have been individually trained. Such cumulative nodes must keep track of which tasks the student is having trouble with, so that if remediation is necessary, it will be to the appropriate task within the selection node. For example, the speed node in Figure 3B should monitor the student's performance on task 1 and task 2 separately, and if the student has trouble with task 1, the remediation target should be task-1-guided-examples. Selection nodes, then, serve two purposes. If the training is currently at the frontier, they cause the training to proceed through the tasks which their children represent in order. If the student requires remediation, selection nodes prevent the training from re-traversing paths of tasks which the student has already mastered that are independent of the task with which the student is currently having problems.

Another issue in developing intelligent tutoring systems for high performance domains is how to divide the phases among specific tasks to be trained. This largely depends on how much overlap of knowledge the tasks contain. For example, in the medical diagnosis domain a student could reach automaticity in taking blood pressure, and not know the first thing about drawing blood samples. The two tasks are very independent of one another; a tree representation might show nodes for all five training phases under each task node. On the other extreme, in the domain of learning to play a piano, one beginning piece requires essentially the same skills as another. The task for playing each piece might not be differentiated at all in a training tree representation. They would merely be different exercises under the same lesson node. Many high performance domains are made up of tasks which are similar enough to share some training phases, but different enough to benefit from some concentrated training on each task.

The result of this design is the use of the classic tree structure representation as a means of formalizing and codifying the curriculum to be taught for a given skill or skill set. Implementation of the tutoring system then involves the design of an interpreter that can utilize, or "parse", the tree-represented curriculum in order to drive the individualized training sequence for a given student. Thus, this is a very general representation scheme in which a wide variety of curriculums aimed at training a high performance skill can be represented.

The expert model for a set of high performance tasks must consist of a means for representing the procedures to be performed and a way of monitoring accuracy and speed. In the automaticity phase of training, accuracy and speed must be monitored for the performance of the primary

task itself, such as MSK manipulation, as well as for performance on the secondary task, such as beep pattern response. The expert model in this ITS design represents an elaboration at each leaf node of the training tree that provides the information concerning expert level performance in the task to be trained at that point in the curriculum. Because the tree representation is so general, and can naturally handle procedural type knowledge, another modified tree structure is used to represent the expertise of actually performing the task or tasks to be trained. The tree structure is used for a hierarchical, as well as ordered, representation of the task, providing structure to the sequence of steps which are represented at the leaf nodes. (see Figure 4). At each level the nodes can be ordered or unordered, depending on the attributes of the procedure to be taught. In addition to providing the expertise needed by the tutoring system to monitor student performance during a training exercise, these procedural structures also provide the expertise needed to randomly generate exercises for the student of the appropriate type at the appropriate time in the training sequence.

Providing a means of representing a student's capabilities is the last major representational and design issue. Student progress consists of working through the training tree and performing adequately on the various tasks appearing at each leaf node. Because the training domain is largely a physically-oriented task with little cognitive load, there is little need for representing misconceptions or bugs. Thus, an overlay model is sufficient. As a result, the same formalism, namely the training tree and the procedural trees, is used to represent the student model, with annotations to indicate status and performance levels on the various phases of training. Evaluating student performance then becomes a search through the tree representing the procedural expertise to compare a student's behavior with that of the expert.

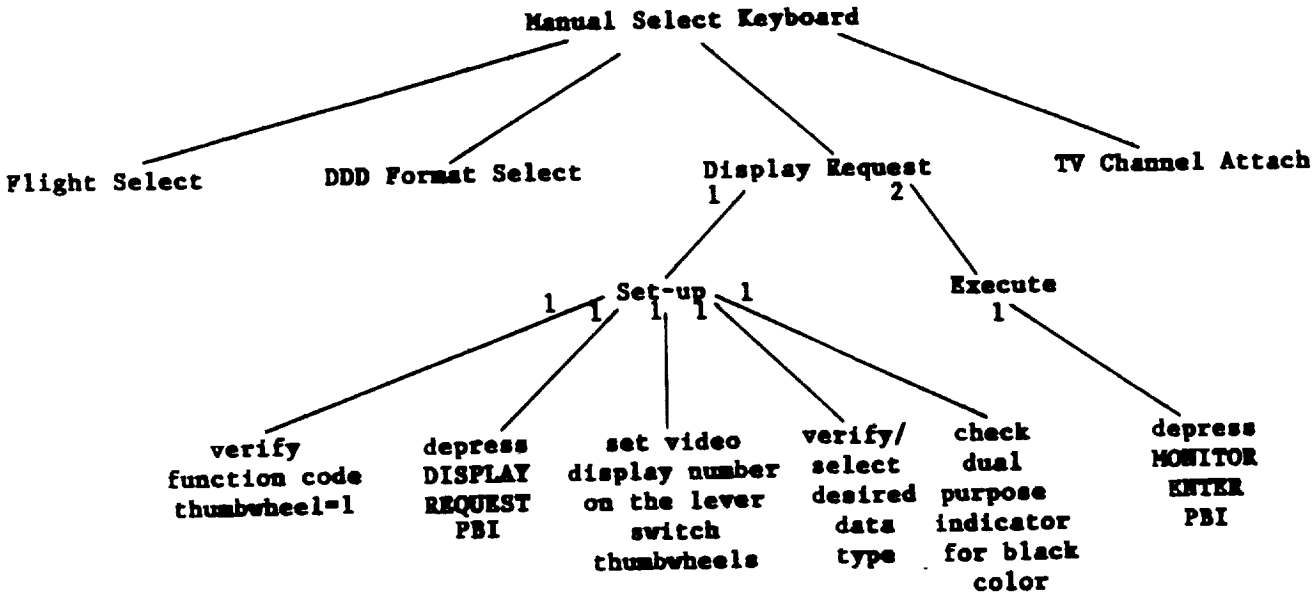


Figure 4: Overview of the DISPLAY REQUEST Procedure Using the MSK

3 THE CONSOLE OPERATIONS TUTOR

The Console Operations Tutor was developed primarily as a research tool for studying the training of high performance skills. However, the system was designed and implemented as though it were to be fielded for training actual shuttle flight controllers for Mission Control at Johnson Space Center. As indicated in the previous discussion, the design of the system was such that a tutoring shell for high performance tasks was developed, for which a training tree was implemented to indicate the desired curriculum, and procedure trees were defined to indicate the tasks to be performed. The high performance task targeted in the Console Operations Tutor was manipulation of the Manual Select Keyboard (MSK). Details of the problem solving domain and the tutor itself are provided in the following sections.

3.1 Space Shuttle Console Operations

Mission Control Center (MCC) at Johnson Space Center (JSC) in Houston, Texas consists of a number of rooms full of computers, video screens, communication networks, large complex consoles, and people, all oriented towards the task of monitoring and controlling shuttle system operations during a flight. A flight controller's job is to monitor a particular portion of the shuttle system through one or more of the complex consoles, using computers and voice communication systems as necessary. An example of a Mission Control Center console for shuttle flight control, namely the front-room propulsion console, is presented in Figure 5. The MCC consoles vary somewhat from one flight function to another, but they generally consist of:

- o one or more video displays (called VDT screens)
- o numerous sets of indicator lights, referred to as Display Decoder Drive Event Lights (called DDD lights)
- o various manual entry devices consisting of numeric thumbwheels, lever switches, and push button indicators including the voice keyset, the manual select keyboard (MSK), the summary message enable keyboard (SMEK), and the display request keyboard (DRK)

as well as one or more other panels for displaying Mission and Greenwich Mean Time. These consoles may also be attached to one or more strip chart recorders for recording sets of analog signals.

In order to become proficient at operating such consoles, flight controllers must learn how to perform such tasks as

- o formatting the various DDD light panels using the MSK
- o selecting, displaying, and reading a variety of video display formats using the MSK, SMEK, and DRK

o selecting and listening to various voice loops using the voice keyset

as well as many others. They must learn to operate these consoles in an automatic manner because such operations are only a means for achieving the goal of ensuring the safe and correct operation of a particular shuttle system, such as the propulsion system, during a space shuttle mission. Should a situation arise where data must be accessed, analyzed, and interpreted, the flight controller must be capable of quickly and effectively accessing the needed data from various video displays and DDD lights without specific thought as to how to manipulate the various keyboards. His/her conscious, cognitive thoughts are too busy diagnosing the situation to be concerned with how to get the data. Thus, console operations can be classified as a high performance task.

The first phase of work on the Console Operations Tutor has centered around training the operation of the Manual Select Keyboard (MSK). This is the keyboard that flight controllers use to initialize the console for the ascent, orbit, and descent phases of a mission. Initialization requires formatting all DDD light panels, selecting several video displays to get information concerning general system status, and selecting various voice loops to listen to monologues and dialogues. Eventually the tutoring system could be expanded to include training on all of the keyboards of a console, as well as a general console overview. Figure 6 provides details of the five-phase training curriculum developed for training the MSK.

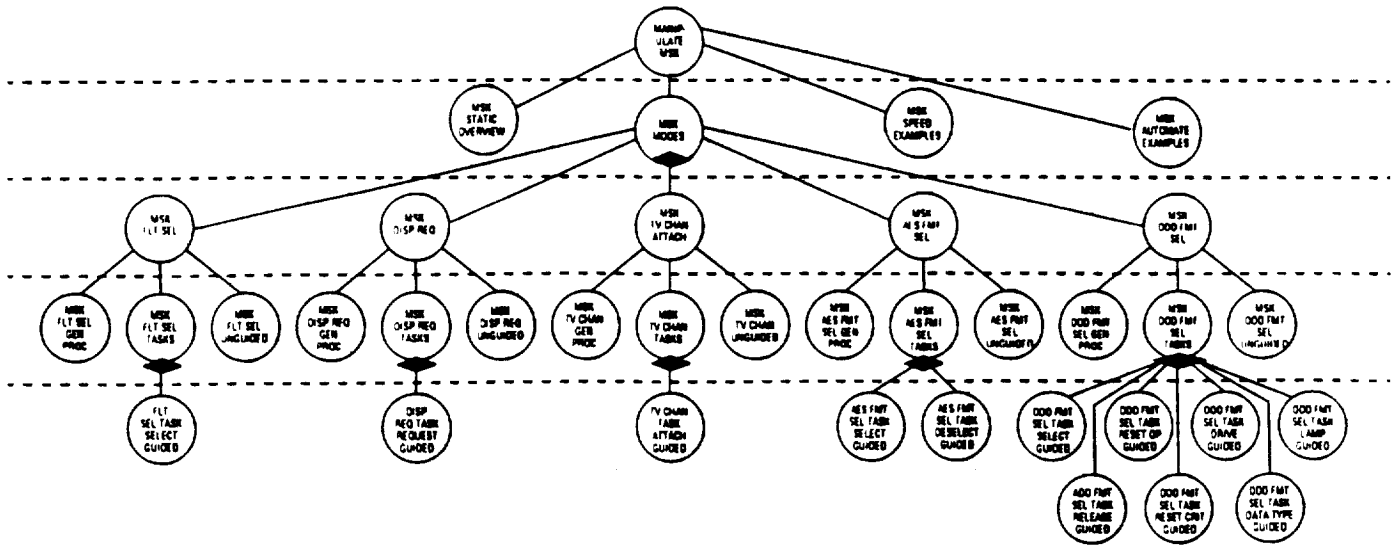


Figure 6: The Training Tree for the Manual Select Keyboard (MSK)

The MSK can be operated in one of five modes. These five modes are indicated by the five push button indicators that appear at the upper right of the panel. The functions vary somewhat from console to console. Operation of the MSK involves first selecting one of these five functions. Selection of the function then defines how the numbers on the thumbwheel and lever switches at the center left of the MSK are interpreted by the Mission Control Computer. These numbers can represent flight numbers, VDT screen numbers, DDD light formats, or codes for performing specific actions (such as reset). Thus, though the manipulation of the MSK only consists of between four and six steps, knowing what buttons to push and how to set the numbers and data types can become quite complex, and difficult to remember.

Currently, flight controllers are trained to run a console through a set of training manuals and workbooks. Then, through a type of apprenticeship, they begin participating in mission simulations, first as observers and then as actual participants. The cognitive load in this arena is extremely high. Individuals are usually degreed in an engineering field related to the system they monitor, and an individual must be trained and certified for each console position s/he runs. As a result, most flight controllers stay within in a single discipline, such as propulsion or navigation. Because there is so much to learn, little time is spent training the task of actually manipulating the console. The skill is acquired through practice during simulations on the actual equipment while other cognitively-oriented tasks are the main issue. Any means of providing cost effective, easily accessible training on the skill of actual console manipulation would help stream-line the training-process for shuttle flight controllers.

3.2 Training Using The Console Operations Tutor

The Console Operations Tutor is implemented in C, CLIPS, and GPR on a color Apollo Domain 4000. These tools were selected for pragmatic reasons. A set of hardware and software tools had to be selected that was readily available to all parties involved in the effort, yet powerful enough to support the CPU- and interface-intensive application. A PC-based system under DOS would have been ideal from the perspective of system availability. However, it would not have been powerful enough for the application. The Apollo Domain provided one of the least expensive color platforms available at the start of the project in 1988. Though little software was available on that platform at that time to simplify interface and intelligent system development, the speed and resource requirements were such that only lower level tools were under consideration anyway. GPR is the basic graphics primitives provided on the Apollo and CLIPS (C Language Integrated Production System) is a C-based, efficient, knowledge-based system development tool designed and implemented at NASA/JSC. Implementation of the Console Operations Tutor began in mid-1988 and the system was delivered in final form to the Armstrong Laboratory Human Resources Directorate and to NASA/JSC in mid-1990.

The display for the tutoring system is organized into five major areas, as illustrated in Figure 7. Across the top third of the screen is a complete graphic representation of the entire console. This provides the

student with an overall layout and organization of the console. The lower left half of the display provides an area where one of the panels from the console can be expanded to provide further detail. The figure shows the MSK panel. The lower center of the screen provides an area for mouse-sensitive buttons used by the student to respond to questions generated by the tutoring system. The lower right half of the screen provides the text interface where the tutor can present information to the student. A top portion of this area, labeled "GOAL" is where exercises are presented to the student. The bottom portion is used for prompting the student and providing feedback.

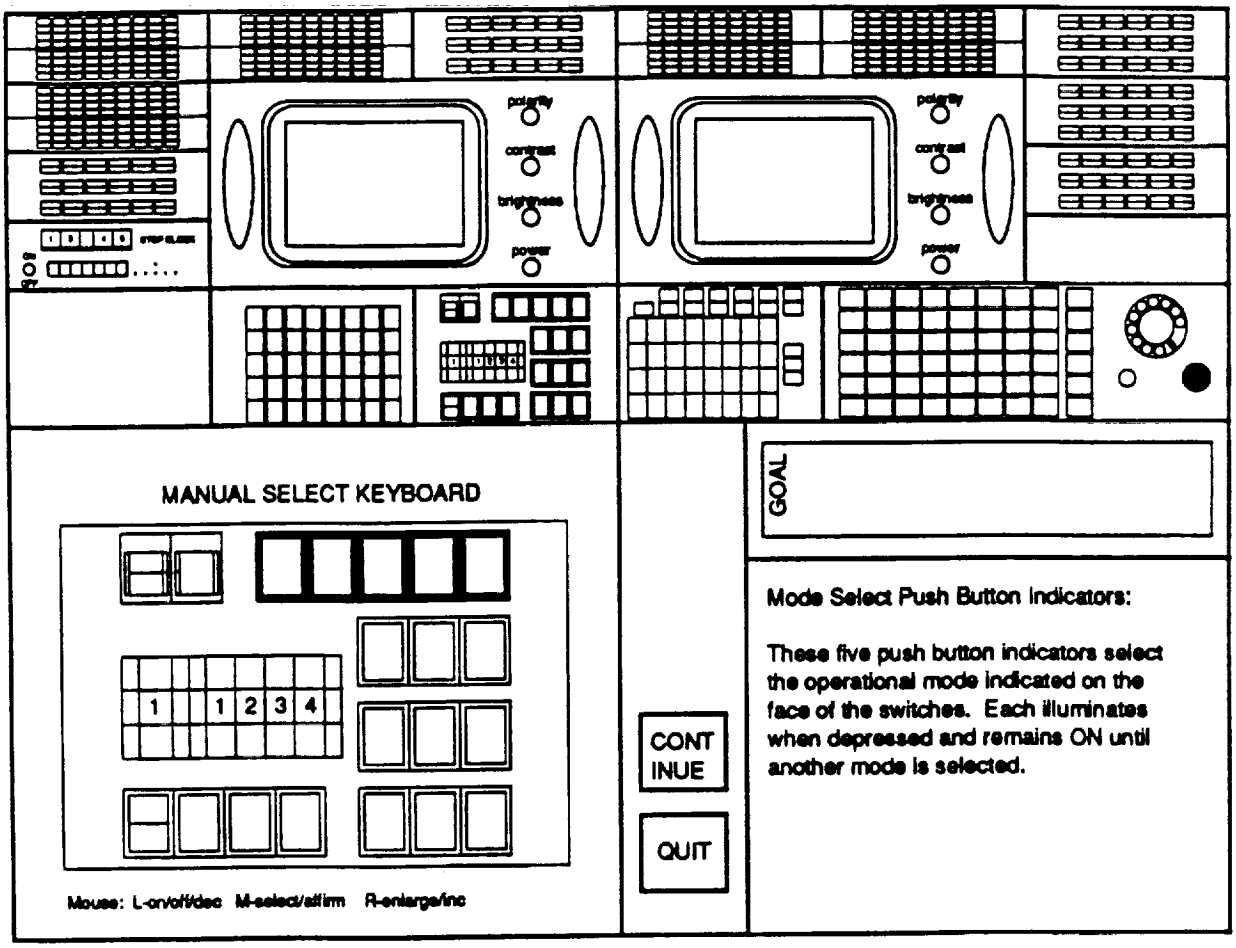


Figure 7: The Student Interface to the Console Operations Tutor

The graphic display of the console is mouse-sensitive. Under certain conditions it allows a student to select panels by clicking over them with the mouse to blow them up in the lower left window of the display. When a panel is expanded and displayed in the lower left window, it too is mouse-sensitive. A student can manipulate it by clicking the mouse over its components, thus incrementing or decrementing a thumbwheel counter, turning a push button indicator on or off, or just getting a display of the text written on the object. In this way, a large portion of the console functionality is simulated graphically and a student can gain experience in performing console operations through these simulated manipulations. The simulation provides high cognitive fidelity, but lower physical fidelity.

Training on the use of the MSK proceeds through the five phases described previously. The first phase of training on the MSK, referred to as "MSK static Overview" (see Figure 6), provides an overview of the MSK layout and structure. The MSK panel is expanded in the lower left window on the screen and the system steps through each of its functional components, highlighting them on the graphics display and describing them with text in the lower right window. This particular phase is illustrated in Figure 7, where the mode select push button indicators are highlighted in the graphics on the left and their description appears in the text on the right. A student can move forward and back at his/her own pace through this portion of the tutorial. At the end, the student must pass an identification test where the student is asked to click over the various components of the console to indicate his/her response to the tutoring system's questions in order to proceed to the next phase of training. Based on the score, the student is allowed to continue to the next phase or required to review the material.

Manipulation of the MSK can take place in one of five modes, selected with the push button indicators in the upper right corner of the MSK panel. The procedure for manipulating the MSK varies depending on the mode selected, so the student's training consists of five general procedures to be mastered. Because all of the objects on the MSK panel remain the same for each procedure, the first static overview phase applies to all procedures. However, at the procedural overview phase, the training tree branches to allow the student to concentrate on learning one of the procedures at a time. A task selection node in the traininig tree, called "MSK modes," is used to select the mode of operation and phases 2 and 3 of training are then subsumed under each independent task nodes in the tree (see Figure 6).

As a result, in the second phase of training, an overview of the procedural process for manipulating the MSK in one single mode, for example display request mode, is given. This is done in a manner similar to the MSK overview. Components are highlighted in procedural order and explanations about each step of the procedure are provided. For example, to request a particular video display to appear in the right monitor of the console, the push button indicator with DISPLAY REQUEST written on it must be pushed, the number of the video display entered on the right four thumbwheels, and the RIGHT MONITOR ENTER push button indicator pressed in the lower right corner of the MSK. A number of other steps must be performed as well, but these are the key steps. The order of all steps does not matter, with the exception that pressing the monitor enter push button indicator must be done last. Thus, the procedure could be

represented as in Figure 4, where five actions comprise an unordered group that constitutes the first step, called "set-up," and the monitor enter action comprises a second step, called "execute." Of course, some sequences are more logical than others and the current system enforces a specific order for performing these actions. In order to move on to the next phase of training, the student must identify each step in the correct sequence.

The third phase of the training, referred to as "guided examples," again concentrates on manipulation of the MSK panel in the same mode as was just presented. Specific examples of the procedure are generated that the system solves as a demonstration for the student. For two of the modes, AES format select and DDD format select, there are multiple specific tasks to be taught. For example, selecting a DDD format and de-selecting a DDD format involve manipulating the same objects in the MSK in the same order, but in slightly different ways. Thus, for these modes the phase three demonstration and practice exercises are repeated for each specific task in sequence (see Figure 6).

The student performs the assigned exercise by manipulating the mouse over the appropriate components of the MSK in the correct order to perform the requested operation. For example, in the guided example phase of training the system may request that the student cause the video display number 12 to appear in the left display monitor. The system will prompt the student at each step and verify its correctness before moving on to the next step. If an error occurs, rules based on the error that the student has just made and the student's history of errors are used to coach the student to perform the procedure correctly. Satisfactory performance in this phase of training is based mainly on accuracy, but speed is also considered. When a student has consistently performed the assigned exercises with complete accuracy and the speed of performance has more or less plateaued, the system allows the student to move on to the next phase of training.

The fourth phase of training involves working on speed and is therefore referred to as "unguided" and "speed" examples. In this phase, the tutoring system no longer guides or coaches the student through the exercises. Instead, the system simply presents an exercise, again concentrating on a single mode of MSK operation at a time, and the student must manipulate the MSK appropriately with the mouse to achieve the requested action. If phase three was repeated for several specific tasks, then this phase acts as a cumulative testing phase, randomly presenting exercises for all of the tasks which the student has learned within this mode of MSK operation. Then, if a student begins making mistakes on a particular type of task, the appropriate MSK mode is selected in the training tree, through the selection node, for remediation.

The static procedure, guided example, and unguided example sequence of phases is repeated for each of the five modes of MSK operation (see Figure 6, the "gen proc", "guided," and "unguided" leaf nodes). The fourth phase of the training wraps up with a cumulative lesson, called "msk speed examples," where tasks from all modes are given in random order for the student to practice. The system at this point watches which modes of MSK operation the student is having trouble with so that, if remediation is necessary, it will be to the appropriate mode. Based on consistently performing with complete accuracy and reaching a point where speed is no

longer improving significantly, the system then allows the student to move on to the final phase of the training.

The fifth and final phase of training is a repeat of the fourth phase only with a secondary task that must be performed simultaneously by the student while doing the assigned exercise. While performing a random selection of MSK operation, the student must also acknowledge certain patterns of beeps by hitting the appropriate function key. The system assumes that the student has successfully automatized the MSK manipulation process when the accuracy in performing both tasks has reached one hundred percent and the speed of performing the assigned exercise and responding to the beeps has reached a peak for that particular student.

It is important to note that during the final three phases of training, where skill is being acquired and tested, no predetermined number of trials is used to determine whether or not the student should move on. Advancement to the next phase in training depends on the particular student's performance. Though accuracy is required to be one hundred percent correct, ultimate speed can vary based on the student. The system determines if speed is reaching the individual's asymptote by comparing speeds on sequential tasks and trends of speeds over time in order to decide when to move on. The decision to backup and review material is based on how much difficulty the student is having attaining the required accuracy and, to a lesser extent, speed. Remediation causes selective backup based on student errors and can even backup all the way to the start of the training program if necessary. In this way the system can be used to refresh the memories of individuals who have been interrupted in their training for a period of time, as well as those who are seeing the material for the first time.

4 FIELDING THE CONSOLE OPERATIONS TUTOR

The fielding of the Console Operations Tutor differed from that of most other AI-based tutoring systems due to the fact that it was delivered to a psychology research laboratory instead of to an actual training environment. The only "users" who had to be satisfied were the psychologists who would be using the system to run experiments and whose goals for the system were, therefore, much different from those of an actual training environment. Thus, the key issues in a successful fielding in the case of the Console Operations Tutor were not so much involved with fitting the system into an existing organization/culture, getting it accepted, and ensuring that it met the predefined training goals, but rather that the system was robust and general enough to handle the changing research goals of the lab. The system had to be capable of holding-up under a stream of non-computer-oriented subjects and of providing the researchers with the data needed to answer their research questions. The following sections describe additional tools and systems provided in fielding the Conole Operations Tutor, and some of the experience gained to date with using the system as a research tool for studying effective training approaches for high performance skills.

Southwest Research Institute made final delivery of the Console

Operations Tutor to the Armstrong Laboratory Human Resources Directorate at Brooks Air Force Base and to NASA/Johnson Space Center in June of 1990. Though the primary reason for developing the tutoring system was to perform psychological research at HRD, the tutor had been developed such that it could be used for actual training at JSC. In fact, a preliminary study at JSC indicates that novice console operators have a high degree of acceptance for the system.

4.1 Console Operations Tutor Experimental Set-Up

The experimental portion of the console operations tutoring system consists of several major components. These are the Console Operations Tutor itself, which is an intelligent tutoring system designed to train the task of MSK manipulation to automaticity, a set of parameters for altering certain attributes to support psychological exerimentation, and a 3-D mock-up of a console with a fully-functional MSK panel that is an exact duplicate of the actual panel found on the consoles in Mission Control. This set-up provides a full array of tools for performing research in high performance skill acquisition, skill transfer, and skill retention.

The Console Operations Tutor contains two facilities that allow the psychologist to manipulate relevant paramenters to study training effects. The first facility allows a researcher to manipulate instructional variables to alter the training between experimental groups and is referred to as the experimental facility. The second facility, the performance facility, collects student performance data for later statistical analysis.

Using the experimental facility, researchers can define the value of 10 instructional parameters. The first three parameters dictate the speed at which a subject must perform in order to be eligible to proceed through the levels of training. The next seven paramters determine the difficulty of the secondary task in automaticity training. The final two parameters determine the phases of training to which the student will be exposed. The following list describes the 10 parameters.

- Maximum Speed - the slowest speed at which a student must be performing consistently to avoid remediation.
- Speed Criterion - the criterion speed that students must reach during the speed phase of training in order to advance to automted training.
- Automate Criterion - the criterion speed that students must reach during the automated phase of training in order to complete training.
- Number of Beep Patterns - the number of alternative beep patterns presented to the student for the secondary task in the automated phase of training.
- Number of Beeps in a Pattern - length of the beep pattern in the secondary task.

- Number of Target Patterns - the number of beep patterns that the student must recognize and respond to on any given exercise on the secondary task in automated training.

- Percentage of Target Beep Patterns - the frequency with which the system provides a target versus a distractor beep pattern to the student on the secondary task.

- Latency between Beep Patterns - the amount of time that elapses between beep patterns during the secondary task.

- Speed Exercise Types - the experimenter may elect to have the student perform any or all of the MSK operations in the speed phase of training.

- Automated Exercise Types - the experimenter may elect to have the student perform any or all of the MSK operations in the automated phase of training.

The performance facility monitors the student's speed and accuracy during the training exercises and writes these indices to a performance file. During each exercise in the guided, unguided/speeded, and automated phases of training, the system monitors the student's level of accuracy. The percentage of accurate steps is written to the performance file. For the unguided/speeded and automated phases of training, the system also collects the speed at which the exercise was performed and writes this measure to the performance file. Finally, during the automated phase of training, the system collects and writes the student's accuracy and speed on the secondary task.

One of the goals of this endeavor was to examine the efficacy of computer-based, high performance skill training by gauging how well a learned skill transfers from the tutor to actual console operations. Because of this goal, a 3-D mockup of the propulsion console was constructed. The mockup is driven by C and CLIPS software running on a 386-based machine. This software presents a problem and the student completes the corresponding MSK operation on a full-fidelity, 3-D mock-up of the MSK panel. The software monitors the student's speed and accuracy and writes it to a file. No feedback is provided to the student since this portion of the tutoring system experimental set-up is concerned with testing performance, not with improving it. The experimenter has the ability to select which of the five MSK operations should be included in the student's test. With this experimental set-up, all of the training and research performed at HRD can be done independent of the availability of consoles in Mission Control at Johnson Space Center.

The tutoring system delivered to Johnson Space Center consisted only of the Console Operations Tutor, with the experimental and performance facilities. No 3-D mock-up was delivered since the goals of the set-up at JSC were more to study the acceptance of this training approach by the flight controllers, rather than the more fundamental question of high performance skill acquisition.

4.2 Initial Results Of The Research Project

The flexibility designed into the Console Operations Tutor is intended to support a series of empirical investigations. The general goal of these investigations is to develop a replicable, fully automated approach to training high-performance tasks. At this time, the first experiment being run at the Armstrong Laboratory Human Resources Directorate is well under way. In the experiment, subjects are trained on the console operations tutor to one of two training criteria. One group of subjects is trained until their accuracy and speed on the task are equivalent to an expert. A second group is trained until they are not only as fast and accurate as an expert, but are also able to perform the secondary task while maintaining that speed and accuracy. That is, the second group is trained until they have developed cognitive automaticity. After training, both groups of subjects are tested on the 3-D mock-up of the console immediately and after delays of 2 weeks, 1 month, and 2 months. The mockup has been engineered to capture performance data during task performance.

The first goal of the initial experiment is to validate or refute the claim that cognitively automated task performance is more reliable, less susceptible to stress, and less susceptible to skill degradation than is task performance which is not cognitively automated. Training time, target task speed, accuracy, and susceptibility to negative interference will be assessed and compared between the two groups for various time delays. The second goal is to demonstrate that a procedure which is cognitively automated in a simulation-based training environment will transfer to an operational environment. Although results of this first experiment are preliminary results are extremely encouraging. Subjects take an average of four hours to achieve the accuracy/speed criteria, as compared to five hours to reach the automaticity criteria. This additional hour of training time is producing large benefits in outcome performance. These benefits include faster and more accurate performance, and an ability to perform under the stress of a secondary task. The complete results of this study will be published in the near future.

Cost to develop this intelligent tutoring system was well under $200,000. The price is nominal compared to the cost to develop a full-fidelity, 3-D simulator to support such training. Preliminary experimental results show effective training that transfers to the actual device can be attained using the less expensive, 2-D intelligent tutoring environment so much may be gained in the use of such systems by organizations requiring high performance skill training.

In addition to the experiment run at HRD, another preliminary study was performed at JSC to examine the issue of transfer and acceptance by actual flight controllers. Subjects were trained on the MSK using the Console Operations Tutor to an automated skill level. These individuals ranged in skill on the actual MSK device from complete novice to fully-trained and experienced flight controller. The novices trained up quickly and found the system very helpful in developing their skill. The already-trained and experienced flight controllers, however, had some difficulty. Their main comment was that they knew what they needed to do, they just had difficulty doing it because the means of doing it, namely the mouse, was so different from the console itself. Thus, though the system

appears to be very useful in training up on a skill that then transfers well, it complicates the task for those already trained in the skill.

5 SUMMARY AND CONCLUSIONS

The Console Operations Tutor was designed and implemented as a research tool to study training approaches for high performance domains. High performance tasks are of increasing interest to organizations such as the Air Force and NASA due to the kinds of jobs they must train. Traditionally, training of such tasks has been ignored, meaning that it takes place on-the-job using real equipment, or it has been a major investment, involving the extensive use of complex, expensive, 3-D simulators. The major research question motivating the development of the Console Operations Tutor is whether or not effective training of a high performance task could take place in a inexpensive, 2-D mock-up using "intelligence" to direct the training. Early results are very positive, thus opening the door to a new, high-potential area of training for intelligent tutoring systems.

The results of the initial studies also suggest that the modified apprenticeship training strategy implemented in the Console Operations Tutor successfully trains high performance skills. The addition of automaticity training to the apprenticeship program, while consuming only an additional hour of time, enhances performance and resistance to skill degredation in the 3-D environment. These results are very encouraging for developing low- cost training programs for a wide variety of high performance skills.

The deployment of the Console Operations Tutor was also very successful. The psychologists who contracted for this specialized research platform have received the system very positively. In fact, they have contracted to extend the training curriculum for ground-based diagnosis of the shuttle propulsion systems in conjunction with training on the MSK. This new research platform will include facilities to assess the effectiveness of joint cognitive and high performance training. Finally, the ratio of cost of development to perceived benefits of the system to the sponsor is very low: the perceived benefits far outweigh the costs.

From a system development perspective, this was a successful intelligent system implementation, involving participants with a wide range of expertise in a truly interdisciplinary effort. The intelligence in the system centers around a tree search algorithm to diagnose student behavior, thereby driving the instruction. The resulting system is essentially a tutoring shell for high performance tasks into which other curriculums and procedures can be entered using the tree-like structures for the training tree and the procedure trees. Intelligent tutoring systems are the frontier of training programs in many areas, and they promise to dramatically reduce training time and cost.

REFERENCES

Anderson, J. R., 1983, The architecture of cognition, Cambridge, MA: Harvard University.

Anderson, J., Farell, R., and Sauers, R., 1984, "Learning to Program in LISP," Cognitive Science, 8, pp. 87-129.

Arons, A.B., 1984, "Computer-based instructional dialogs in science course," Science, Vol. 224, No. 4653, June, pp. 1051-1056.

Bloom, B.S., 1956, "Taxonomy of educational objectives: The classification of educational goals," in B.S. Bloom (ed.), Cognitive Domain (Handbook 1), New York: McKay.

Bloom, B.S., 1984, "The Two-Sigma Problem: The Search for Method of Group Instruction as Effective as One-to-One Tutoring," Educational Research, June-July, pp. 3-15.

Brown, J.S., Burton, R.R., and deKleer, J., 1982, "Knowledge Engineering and Pedagogical Techniques in SOPHIE I, II, and III," in D. Sleeman and J.S. Brown (eds.), Intelligent Tutoring Systems, London: Academic Press.

Carbonell, J.R., 1970, "AI in CAI: An Artificial Intelligence Approach to Computer-Aided Instruction," IEEE Transactions on Man-Machine Systems, MMS-11(4), pp. 190-202.

Chase, W. G., and Ericsson, K. G., 1981, "Skilled memory," in J. R. Anderson (ed.), Cognitive skills and their acquisition, Lawrence Erlbaum: Hillsdale, NJ.

Carroll, J., 1963, "A model of school learning," Teachers College Record, 64, pp. 723-733.

Cohen, P.A., Kulik, J., and Kulik, C.C., 1982, "Educational outcomes of tutoring: A meta-analysis of findings," American Educational Research Journal, 19(2), pp. 237-248.

Colle, H. A., and Demaio, J., 1978, "Measurement of attentional capacity load using dual-task performance of operating curves," (Interim Rep. AFHRL-TR-78-5), Brooks AFB, TX: Air Force Systems Command.

Collins, A., Brown, J.S., and Newman, S.E., 1987, "Cognitive Apprenticeships: Teaching the Craft of Reading, Writing and Mathematics," in L.B. Resnick (ed.), Cognition and Instruction: Issues and Agendas, Lawrence Erlbaum: Hillsdale, NJ.

Corno, L., and Snow, R.E., 1985, "Adapting teaching to individual differences among learners," in M.C. Wittrock (ed.), Handbook of Research on Teaching (Third Edition), New York: Macmillan.

Cronbach, L.J., and Snow, R.E., 1977, Aptitudes and instructional methods: A handbook for research on interactions, New York: Irvington.

Dede, C., and Swigger, K., 1987, "The evolution of instructional design principles for intelligent computer-assisted instruction," Proceedings of the American Educational Research Association.

Fink, P.K., in press, "The Role of Domain Knowledge in an Intelligent Tutoring System," in H. Burns, C. Redfield, and J. Parlett (eds.), Intelligent Tutoring Systems: Dimensions of Design, Lawrence Erlbaum: Hillsdale, NJ.

Fisk, A. D., and Schneider, W., 1983, "Category and word search; Generalizing search principles to complex processing," Journal of Experimental Psychology: Learning, Memory, and Cognition, 9, pp. 177-195.

Gott, S.P., 1988, "Apprenticeship Instruction for Real-World Tasks: The Coordination of Procedures, Mental Models, and Strategies," in E.V. Rothkopf (ed.), Review of Research in Education, 15, American Educational Research Association: Washington, D.C.

Hancock, P. A., and Pierce, J. O., 1984, "Toward an attentional theory of performance under stress: Evidence from studies of vigilance in heat and cold," in A. Mital (ed.), Trends in ergonomics/human factors I, New York: North Holland.

Kristofferson, M. W., 1972, "Effects of practice on character classification performance," Canadian Journal of Psychology, 26, pp. 54-60.

Lewis, M.W., McArthur, D., Stasz, C., and Zmuidzinas, M., 1990, "Discovery-based tutoring in mathematics," Working notes: AAAI spring symposium series, Stanford University, Stanford, CA.

Newell, A. and Rosenbloom, P., 1981, "Mechanisms of skill acquisition and the law of practice," in J. R. Anderson (ed.), Cognitive Skills and Their Acquisition, Hillsdale, N.J.: Erlbaum Associates.

Regian, J.W., 1989, "Representing and Teaching High Performance Tasks Within Intelligent Tutoring Systems," Proceedings of the Second Intelligent Tutoring Systems Research Forum, San Antonio, TX, pp. 11-22.

Regian, J.W., and Shute, V.J., 1988, "AI in Training: The Evolution of Intelligent Tutoring Systems," Proceedings of the Conference on Technology and Training in Education, Biloxi, MS.

Schneider, W., and Shiffrin, R. M., 1977, "Controlled and automatic human information processing: I. Detection, search, and attention," Psychological Review, 84, pp. 1-66.

Shiffrin, R. M., and Schneider, W., 1977, "Controlled and automatic human information processing: II. Perceptual learning, automatic attending, and a general theory," Psychological Review, 84, pp. 127-190.

Shute, V.J., in press, "Aptitude-treatment interactions and cognitive skill

diagnosis," in J.W. Regian and V.J. Shute (eds.), Cognitive approaches to automated instruction, Hillsdale, NJ: Lawrence Earlbaum Associates.

Sleeman, D.H., and Brown, J.S. (eds.), 1982, Intelligent Tutoring Systems, Academic Press: London.

Woolf, B., and McDonald, D.D., 1985, "Building a Computer Tutor: Design Issues," AEDS Monitor, 23(9-10), pp. 10-18.

Wenger, E., 1987, Artificial intelligence and tutoring systems, Los Altos, CA: Morgan Kaufman.

Woolf, B.P., 1987, "A survey of intelligent tutoring systems," Proceedings of the Northeast Artificial Intelligence Consortium, Blue Mountain Lake, NY.